Eric Santos

Executive Summary

The purpose of this project is to generate groups and a bracket for a tournament. It divides all teams into groups of 4, generates a schedule with games home and away, and creates a bracket for the tournament. Groups can be refreshed to account for results in the schedule and the bracket automatically updates with the winner. I created this to help my friends and I organize our soccer tournaments.

Implementation

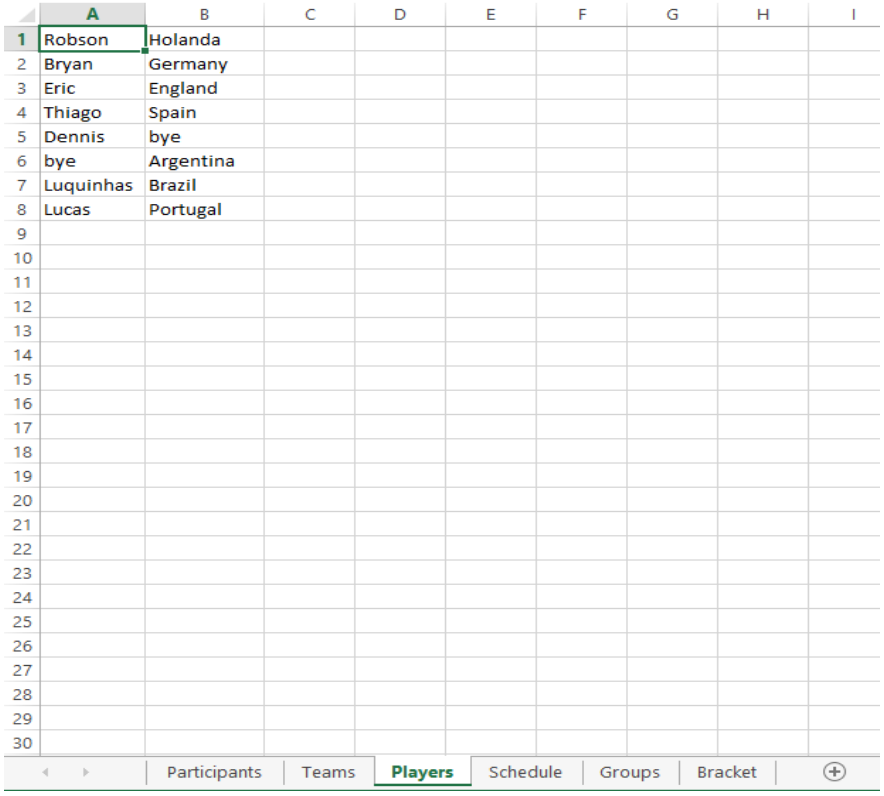
To implement this project, I created a Class Module called CTournament that contains all methods necessary to determine which player gets which team, create the groups, organize the schedule using the round robin algorithm, and lay out the bracket with formulas to calculate the winner according to the results.

This class also knows how to order each group according to the results in the schedule sheet.

I will give a detailed description of each of these sections below.

1 – Determine teams

There are three sheets relative to this part of the code. One called Participants, one Teams and another Players. The Participants sheet is meant to put the real names of people in the tournament. The Teams sheet needs to have the names of the teams that will be in the tournament. And the Players sheet will be populated with a random assignment of each participant to a team. This will make the tournament fair.



2 – Determine groups

I keep multi-dimensional array called “groups” that help me keep track of which team is in what group. I will use this array to generate my schedule of games (see below). Groups are kept in memory until displayed in the “Groups” tab.

3 – Create Schedule

The schedule was one of the more challenging parts in this project. I not only had to make sure each team played against all other teams, but I also had to find a way to organize it in rounds, so that I could create an actual schedule out of it.

After much Googling (and possibly some Binging) and a lot of failed attempts to generate my schedule (my first few initiatives only worked for the first few games and would repeat at least one game per round), I found some good explanation about the Round Robin algorithm and decided to implement that.

```
' Get the second half of teams firts
' then get the other half in reverse order excluding first element
Dim l As Integer
Dim index As Integer
index = 1
For l = (numberOfTeams / 2) + 1 To numberOfTeams
    roundRobinArray(index) = groupArray(l)
    index = index + 1
Next l

For l = (numberOfTeams / 2) To 2 Step -1
    roundRobinArray(index) = groupArray(l)
    index = index + 1
Next l

Dim m As Integer

index = 1

For l = 1 To numberOfRounds
    combination(index, 1) = roundRobinArray((l Mod UBound(roundRobinArray)) + 1)
    combination(index, 2) = groupArray(1)
    index = index + 1
    For m = 2 To ((UBound(groupArray) / 2))
        Dim firstTeam As Integer
        firstTeam = ((l + m) Mod UBound(roundRobinArray) + 1)

        Dim secondTeam As Integer
        secondTeam = ((l + UBound(roundRobinArray) - m) Mod UBound(roundRobinArray) + 1)

        combination(index, 1) = roundRobinArray(firstTeam)
        combination(index, 2) = roundRobinArray(secondTeam)
        index = index + 1
    Next m
Next l
' END ROUND ROBIN!
```

It was hard to implement this algorithm and make it work with the variables and arrays I already had set up, but it totally paid off, since I could not have come up with that myself. After the piece of code above runs, my combination() array contains a list of all possible games for a certain group. Notice that I don't have a list for all games in the tournament, just for this one group. (Which means I loop and do this for each group in my groups array).

4 – Populate Schedule

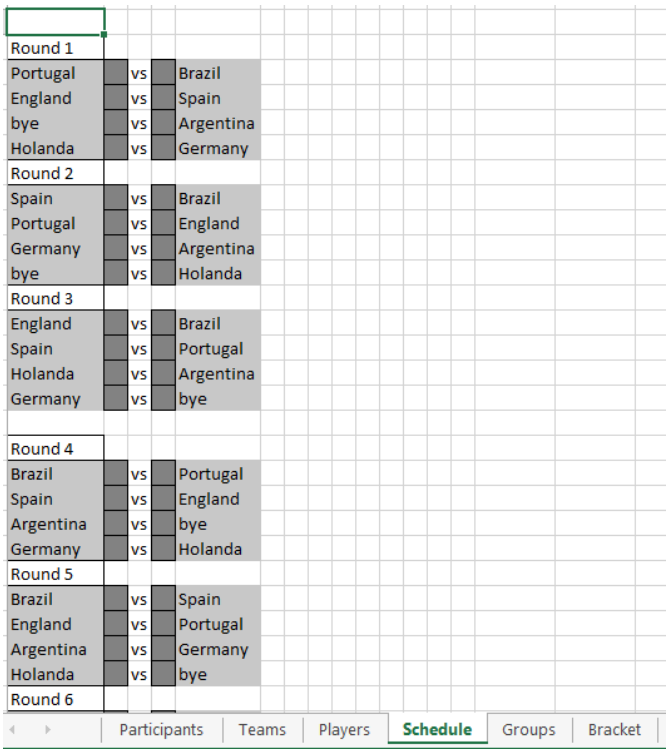
The next step is to actually display this array in the "Schedule" sheet in a way that the user can easily see what games are happening in each round. The main challenge in this part was to figure out the exact position of the games. Remember, I only had one group at a time, so I had to fill up the schedule for the group leaving enough room for the next group's games, in my loop.

Here is what I did:

```
' the index identifies how far I am as far as the groups go. 1 means we are doing the first group, 2 the second and
index = 1
For i = 1 To numberOfRounds

    gamesPerRoundPerGroup = UBound(combination) / numberOfRounds

    gamesPerRound = gamesPerRoundPerGroup * UBound(groups)

    ActiveCell.Offset(numberOfGroups * gamesPerRoundPerGroup * (i - 1) + i, 0).Value = "Round " & i
    ActiveCell.Offset(numberOfGroups * gamesPerRoundPerGroup * (i - 1) + i, 0).Borders().ColorIndex = 1

    Dim x As Integer
    ' the counter helps me know how many of the current games I have populated
    Dim counter As Integer
    counter = 1
    For x = index To gamesPerRoundPerGroup * i
        With ActiveCell.Offset((i - 1) * gamesPerRound + (counter + (gamesPerRoundPerGroup * (J - 1))) + i, 0)
            .Value = combination(x, 1)
            .Interior.Color = RGB(200, 200, 200)
            .Offset(0, 1).Interior.Color = RGB(130, 130, 130)
            .Offset(0, 1).Borders().ColorIndex = 1
        End With
        ActiveCell.Offset((i - 1) * gamesPerRound + (counter + (gamesPerRoundPerGroup * (J - 1))) + i, 2) = "vs"
        With ActiveCell.Offset((i - 1) * gamesPerRound + (counter + (gamesPerRoundPerGroup * (J - 1))) + i, 4)
            .Value = combination(x, 2)
            .Interior.Color = RGB(200, 200, 200)
            .Offset(0, -1).Interior.Color = RGB(130, 130, 130)
            .Offset(0, -1).Borders().ColorIndex = 1
        End With
        index = index + 1
        counter = counter + 1
    Next x
Next i
```

Notice that I use ActiveCell.Offset for everything, so that I could be flexible enough to support different types of group sizes and etc. This was very challenging to get just right. Basically I had to figure out in what round I am (to know how many rows to jump ahead) and how far into the group I have been (so that within the same round, I know how many rows to skip). After the schedule is ready, we can go ahead and populate the groups.

Here is how a schedule looks like:



Round 1				
Portugal		vs		Brazil
England		vs		Spain
bye		vs		Argentina
Holanda		vs		Germany
Round 2				
Spain		vs		Brazil
Portugal		vs		England
Germany		vs		Argentina
bye		vs		Holanda
Round 3				
England		vs		Brazil
Spain		vs		Portugal
Holanda		vs		Argentina
Germany		vs		bye
Round 4				
Brazil		vs		Portugal
Spain		vs		England
Argentina		vs		bye
Germany		vs		Holanda
Round 5				
Brazil		vs		Spain
England		vs		Portugal
Argentina		vs		Germany
Holanda		vs		bye
Round 6				

Participants | Teams | Players | **Schedule** | Groups | Bracket

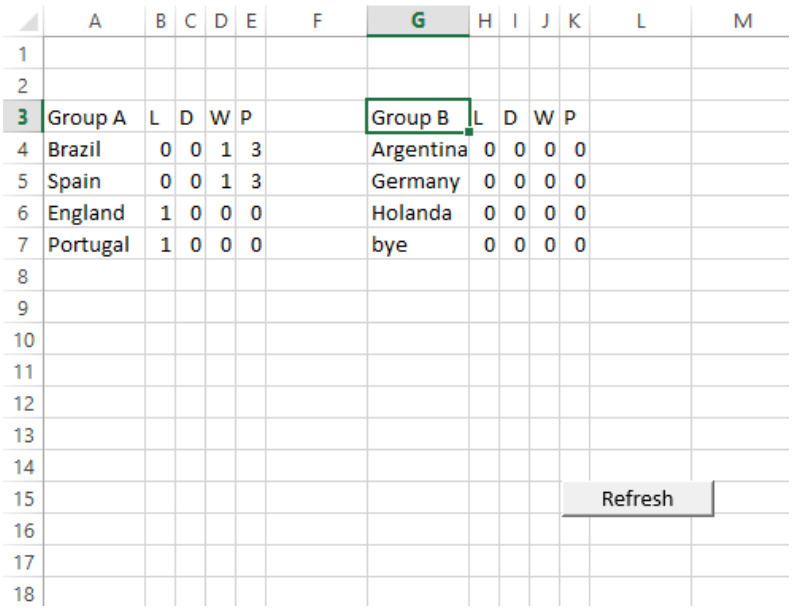
5 – Populate Groups

Displaying the groups is also challenging because I had to position things just right. Using Offset and some control variables gets very confusing sometimes, I was able to use some control variables to assist me in displaying the groups in two columns rather than only one.

This is how the groups tab looks like:

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3	Group A	L	D	W	P		Group B	L	D	W	P	
4	Brazil	0	0	0	0		Argentina	0	0	0	0	
5	Spain	0	0	0	0		Germany	0	0	0	0	
6	England	0	0	0	0		Holanda	0	0	0	0	
7	Portugal	0	0	0	0		bye	0	0	0	0	
8												
9												
10												
11												
12												
13												
14												
15											Refresh	
16												

The refresh button is one of the most involved ones. The idea behind is that people will fill out the schedule tab with the results for each game, and by clicking “Refresh” they can see those results reflect in the group. (For example, if Brazil beat Spain in the first game in the schedule, this is what the group should look like:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2													
3	Group A	L	D	W	P		Group B	L	D	W	P		
4	Brazil	0	0	1	3		Argentina	0	0	0	0		
5	Spain	0	0	1	3		Germany	0	0	0	0		
6	England	1	0	0	0		Holanda	0	0	0	0		
7	Portugal	1	0	0	0		bye	0	0	0	0		
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													

To assist me in calculating the group order, I created a orderedGroup array and a calculateGroupOrder function that returns an ordered array with all the values ready to repopulate the Groups sheet. The logic is not very complicated. I had to loop through each round in my tournament, then each game per group in the round, and find where exactly the results are. If the results are empty, then the game has not happened yet and I don’t have to update my array. Otherwise, I give the winning team 3 points (1 if there was a draw), and update the statistics of both teams.

After the calculations, I have to make sure that the array is ordered by number of points. For that I implemented a simple bubble sort algorithm that does the job. Here it is:

```
For i = LBound(orderedGroup) To UBound(orderedGroup)
    For x = i To UBound(orderedGroup)
        If (orderedGroup(x, 5) > orderedGroup(i, 5)) Then
            Dim row(1 To 5) As String
            ' swap
            row(1) = orderedGroup(x, 1)
            row(2) = orderedGroup(x, 2)
            row(3) = orderedGroup(x, 3)
            row(4) = orderedGroup(x, 4)
            row(5) = orderedGroup(x, 5)

            orderedGroup(x, 1) = orderedGroup(i, 1)
            orderedGroup(x, 2) = orderedGroup(i, 2)
            orderedGroup(x, 3) = orderedGroup(i, 3)
            orderedGroup(x, 4) = orderedGroup(i, 4)
            orderedGroup(x, 5) = orderedGroup(i, 5)

            orderedGroup(i, 1) = row(1)
            orderedGroup(i, 2) = row(2)
            orderedGroup(i, 3) = row(3)
            orderedGroup(i, 4) = row(4)
            orderedGroup(i, 5) = row(5)
        End If
    Next x
Next i
```

After all this, I have an array with the group information ordered by number of points.

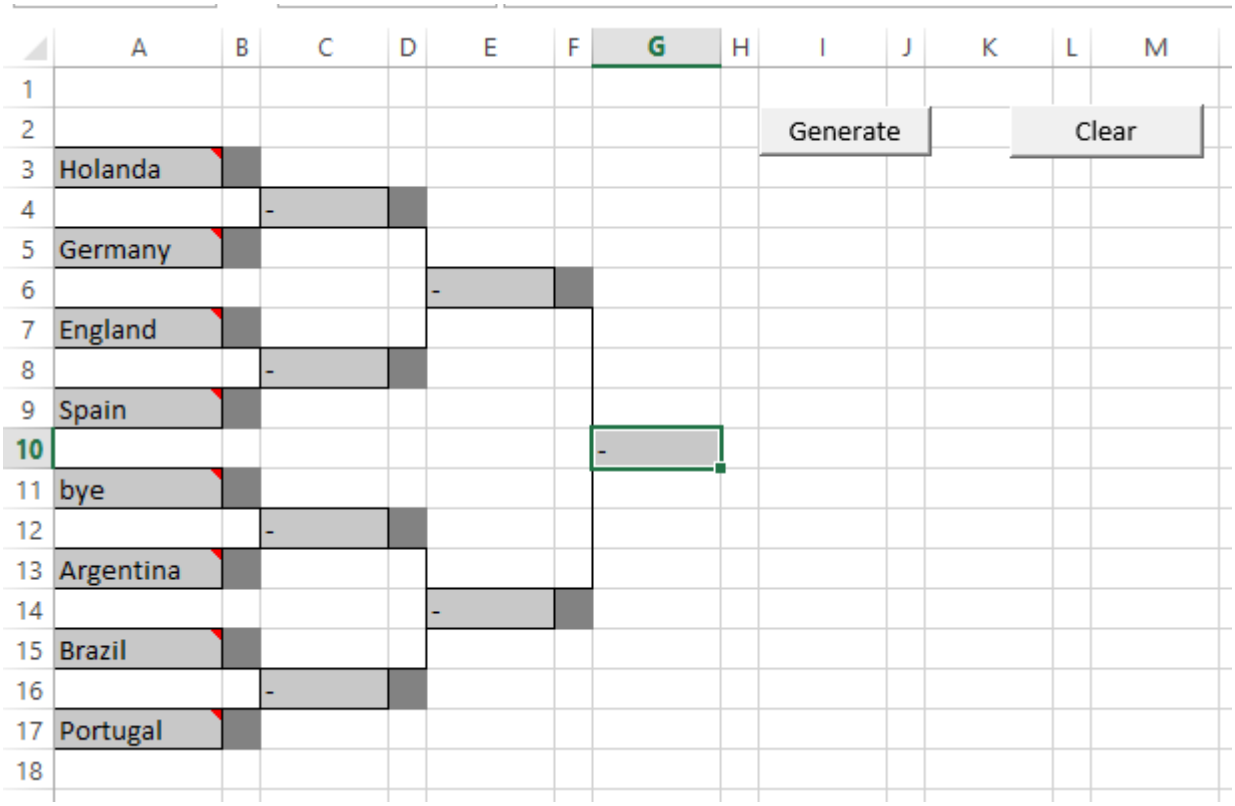
This way, when the user hits refresh, the PopulateGroup function will call the CreateOrderedGroup function before displaying each group.

6 – Create the Tournament

The tournament was probably the hardest one to figure out. There is a lot of math behind the pattern on a tournament schedule, and I had to dig into how, mathematically, I could generate a tournament that displays things nicely organized when I have a variable number of teams in my input.

As a disclaimer, my bracket generator is working well when the appropriate number of teams is placed in the tournament. If I have a number that is a power of 2, it works fine. The problem is when I have odd numbers with which it is not possible to lay out a perfect tournament unless another criteria is determined (do we have qualifiers? Some teams will get a bye round?). Since I have already spent too much time in this project, I decided not to implement these extra features right now.

This is how my tournament tab looks like:



Now, it may look somewhat easy to lay this out, but it was actually quite challenging. For each round of the tournament I had to figure out how much spacing in between the games I should have (which changes according to the number of teams and etc) as well as how to create nice borders to give it a feel that two games connect to one game in the following round.

Most importantly, I also implemented the functionality that when games results are typed in, the winning team would be automatically populated into the next round's cell, so the user doesn't have to type the winner's name.

If you look at the populateBracket function, I have to get the Log base 2 of the number of teams I have left in the tournament to determine how many rounds I have left. As I went through each round, I noticed that the spacing between games increased in this pattern: 1 row between games for the first round, 3 rows between games for the second round, 7 rows between games for the third round, and etc. I realized that this was the sum of the all powers of 2, starting at zero. For example, at round one, the spacing was 2 to the power of 0. At round 2, it was 2 to the power of 0 plus 2 to the power of 1. At round 3, It was the sum of 2 to the power of 0, 2 to the power of 1 and 2 to the power of 2.

So I decided to create a helper function that calculates the sum of the powers of 2, and all I had to do give it my current round and it would return the spacing I needed to do between each game.

This sounds somewhat straight forward now, but believe me, I spent a long time trying to figure this out.

I also added comments in the first cells so that the user can quickly see what person is playing with that specific team.

What I learned

This project helped me learn some concepts that we have not touched in class and reinforce others that we have discussed in class.

I think the most useful thing I learned was how to use a Class Module. I liked that I could create a tournament object and call my own functions through it. I see how class modules can help organize and structure VBA code, so I am glad I tried using it.

I learned a good deal about just using VBA overall. By debugging weird errors I was getting, I realized that I used my variables in the wrong way. I learned that you have to have a static number at compile time for your variable dimensions, but that I can redim it using a variable.

I learned that it isn't easy to lay things out relatively to one another using one loop that fits all scenarios. Though sometimes it is faster just to hard code things, I enjoyed the challenge of figuring out specific patterns and algorithms that would provide the result I needed.

This project also helped me get some more experience with ordering and shuffling arrays, and I also learned how to add comments to cells. I had never done that before.

Unfortunately, I underestimated the amount of work this project was going to take. Some of the problems I was trying to solve took so long that I didn't have time to implement all the features I wanted. My original plan was to account for different types of tournaments and to handle the edge case scenarios (when the number of teams is not perfect) in a good way.

Assistance

I had not assistance as far as no one typed any of my code for me. I did however, use the internet a lot to try to find solutions for my problems. For some of the algorithms I have (like Round Robin), I found an implementation online in a different programming language and I had to port it to VBA and make it work with my program. So Google helped me, but at the end of the day I implemented everything.